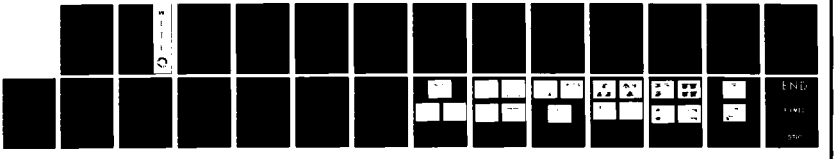
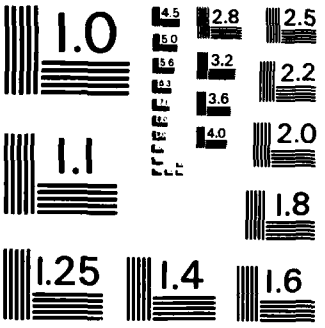
AD-A160 586 IMAGE PROCESSING FOR VISUAL NAVIGATION OF ROADWAYS(U) 1/1
MARYLAND UNIV COLLEGE PARK CENTER FOR AUTOMATION
RESEARCH J LEMOIGNE ET AL. SEP 85 CAR-TR-138 ETL-0406
UNCLASSIFIED DACA76-84-C-0004 F/G 17/7 NL





ETL – 0406

(2)

AD-A160 586

Image processing for visual navigation of roadways

Jacqueline LeMoigne
Allen M. Waxman
Babu Srinivasan
Matti Pietikainen

Center for Automation Research
University of Maryland
College Park, Maryland 20742

September 1985

Prepared for
U.S. ARMY CORPS OF ENGINEERS
ENGINEER TOPOGRAPHIC LABORATORIES
FORT BELVOIR, VIRGINIA 22060 – 5546

E T L

Destroy this report when no longer needed.
Do not return it to the originator.

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

The citation in this report of trade names of commercially available products does not constitute official endorsement or approval of the use of such products.

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: ETL-0406	2. GOVT ACCESSION NO.: AD-A166 586
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): IMAGE PROCESSING FOR VISUAL NAVIGATION OF ROADWAYS	5. TYPE OF REPORT & PERIOD COVERED: Technical Report, July 1984 - July 1985
	6. PERFORMING ORG. REPORT NUMBER: CAR-TR-138, CS-TR-1536
7. AUTHOR(s): Jacqueline LeMoigne, Babu Srinivasan, Allen M. Waxman, Matti Pietikainen	8. CONTRACT OR GRANT NUMBER(s): DACA76-84-C-0004
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Center for Automation Research, University of Maryland, College Park, Maryland 20742	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: U.S. Army Engineer Topographic Laboratories, Fort Belvoir, Virginia 22060-5546	12. REPORT DATE: September 1985
	13. NUMBER OF PAGES: 23
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Image processing
Computer vision
Road following
Linear feature extraction
Autonomous navigation
Region-based segmentation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Computer Vision Laboratory at the University of Maryland has developed several image processing algorithms for use by an autonomous land vehicle in order to navigate roadways. This report describes one module of this system, the Image Processing module, which extracts 2-D symbolics from the imagery to be analyzed in the world domain.

PREFACE

This document was prepared under contract number DACA76–84–C–0004 for the U.S. Army Engineer Topographic Laboratories, Fort Belvoir, Virginia by the Center for Automation Research, University of Maryland, College Park, Maryland. The Contracting Officer's Representative was Ms. Rosalene M. Holecheck.

A-1

CONTENTS

1. INTRODUCTION

In support of the Autonomous Land Vehicle (ALV) project, which is a part of the Strategic Computing Program developed by DARPA [1], the Computer Vision Laboratory of the University of Maryland has designed a visual navigation system which will enable the ALV to perform road following tasks. Our early efforts on a general framework for visual navigation were presented in [2,3]; it is based on the decomposition of visual navigation tasks into three levels of resolution, called *long*, *intermediate* and *short range navigation*. Road following is an example of *intermediate range navigation*. The task is to compute a path in an *intermediate range* environment through a region of uniform visibility/navigability established first by long range navigation. This path can be defined as a *corridor of free space*. A corridor of free space is a swath of navigable terrain that is not so densely populated with obstacles that the vehicle could not maneuver among them. Roads are examples of such corridors of free space.

We have developed a modular system architecture to perform visual navigation in general, with a flow of control to support the "road following task" (cf. Fig.1). This system is decribed in detail in [4]. It consists primarily of three Vision Modules which embody image processing tools, geometrical and model-based constraints, and rule-based reasoning capabilities; it is the responsibility of these modules to establish a representation of the three-dimensional structure of the road scene. This system also includes a Planner, a Navigator and a Pilot. The Planner establishes goals and operating constraints for the Navigator, which is responsible for generalized path planning. The Pilot interprets this path into basic commands understandable by the ALV's motor controllers. The Vision Executive is the module responsible for the overall vision process control. It is also the Vision Modules and Executive which establish a labeling of the scene, consistent with :

- the representations derived from the image domain (2-D symbolics)
- the geometrical interpretations
- the knowledge base

Thus, the Image Processing module converts the imagery to 2-D symbolics in the image domain, the Geometry module tries to place these symbols in the 3-D world domain, and the rule-based Reasoning module labels these symbols in a consistent fashion in order to create a representation of the road scene for the navigator. This paper describes in detail the Image Processing module and the 2-D symbolics which are used to construct a 3-D interpretation.

Many symbolic representations can be extracted from the image domain data. Therefore, the Image Processing module is like a toolbox of procedures. Its objective is to establish a variety of independent, symbolic representations in the image domain of the TV imagery itself. It attempts to extract

dominant linear features to provide a boundary-based representation, and segment the grey-level or color imagery to provide a region-based representation. A mature system will combine a number of such representations in order to interpret the scene.

We also distinguish between a *bootstrap phase* and a *feed-forward phase.* Bootstrapping implies no prior visual processing of relevance is available to lighten the current load of visual processing. It corresponds to the vehicle suddenly "opening its eyes" and looking for the road in order to orient itself properly and begin its travel. Feed-forward implies a predictive capability in which visual processing of a prior road segment, taken together with a dead-reckoning capability, is used to predict the approximate locations of important road features in a subsequent image; that is, a focus of visual attention is provided.

2. BOOTSTRAP IMAGE PROCESSING

When the ALV begins its road following task, as no information is usually available about where the vehicle is with respect to the road, it must process the entire image in order to construct a model of the visible portion of road, in the world domain. This is the bootstrapping phase.

2.1. Linear feature extraction

We have developed image processing algorithms which extract the *dominant linear features* in the imagery under the assumption that roadways generate such features in a scene. We are not interested in all the linear features in a scene, only the dominant ones. These features are usually correlated with road boundaries, markings and shoulders, all of which are important for navigating on roads.

In both modes, bootstrap and feed-forward, the extraction of linear features is the result of a sequence of image processing steps including smoothing, gradient, extraction of dominant directions and Hough transformation in order to find the line segments corresponding to these principal directions. For the bootstrap mode, we describe these steps in more detail in the context of the example, "BENDING ROAD", as illustrated in Figures 2.

The various processing steps applied over the image are:

(1) The original image is shown in Figure 2a. The image is first blurred to reduce the noise, by performing a local average in a 5×5 neighborhood.

(2) The Sobel operator is then applied to obtain the gradient magnitude and direction at each pixel. The magnitude of the gradient, encoded in grey, is shown in the upper left quadrant of Figure 2b. The upper right quadrant of the same picture shows the distribution of directions of the gradient vector (encoded in grey) wherever the magnitude exceeded a very low threshold (to remove regions of constant grey level from consideration). Dominant linear features in the original correspond to long features of constant grey level in this direction picture.

(3) The next step is to enhance the linear features in the image. Our enhancement method utilizes the gradient orientation as the dominant measure. The strength of an edge depends upon the edge orientations of neighboring pixels: this is the notion of "local support". For a measure of this local support, we first choose a neighborhood around the considered pixel: it is a rectangular window centered around this pixel, oriented approximately in the same direction as the center pixel gradient direction. A *count* is then calculated, and assigned to the center pixel. Each pixel within the window contributes either a 0 or 1 to the count:

If the direction of an outer pixel is within 15 degrees of the value

corresponding to the center pixel, then the outer pixel contributes a 1

Else, it contributes a 0.

Therefore the percentage of pixels that contributed a 1 represents the amount of local support for an edge at this pixel. The enhanced image in the lower left of Figure 2b illustrates small neighborhoods in which the gradient direction is fairly constant. The support provided in a local neighborhood is encoded in grey.

(4) Following the remark made in (2), our goal is to extract the long features of constant direction. In order to do so, we compute a weighted histogram of the direction image. This is essentially a direction distribution, weighted on the basis of gradient magnitude and local support for linear features. This weighted histogram is shown in the lower right of Figure 2b. Its range is from 0 to 180 degrees, in 127 bins, with the right and left ends connected (i.e. the sense of the direction is ignored).

(5) Peaks in this histogram correspond to directions of potentially long, linear features. After performing a Gaussian smoothing with an effective spread of 11 bins, the next step is to automatically extract the peaks of this histogram. This is done by first finding the local maxima and minima in the smoothed histogram, then a merit value is assigned to each peak, and the peaks are ordered on that basis. The merit function takes into account:

The height of the peak, (i.e., the number of "weighted pixels" which have created this peak)

The "contrast" of that peak compared to the "background" of the histogram, which compares the local maximum corresponding to a peak to the two closest local minima on each side of it.

The merit function adopted is : $\frac{height\ of\ maximum^2}{mean\ of\ two\ minima}$

The positions of the pixels in the original image which contribute to each peak are then determined, and stored as binary pictures. These *peak pictures* will have, for the strong peaks, a cluster of points that delineate dominant linear features. The points corresponding to the strongest four peaks are shown in Figure 2c.

(6) The "peak-pictures" created at the previous step are usually prone to a "salt and pepper" kind of noise. Therefore, we try to isolate the clusters by performing a (4-connected) "shrink and expand" procedure on each peak-picture. This shrinks away the isolated points but leaves clusters unaffected. The results of this procedure are shown in Figure 2d.

An interesting detail of this processing is the kind of data structure

utilized for this "shrink and expand" procedure. Since the *peak picture* for each peak is represented by a binary picture, up to eight peaks can be stored in a byte picture (eight bits per pixel). The use of this data structure not only saves an enormous amount of memory, but also significantly reduces processing time since the "shrink and expand" operation can now be done in parallel for all peaks:

The *shrink* operation does not affect image points having value zero. Using the data structure mentioned above, as the ranges occupied by the peaks are mutually exclusive, at most a single bit in the byte picture can have a value of one. Therefore a single pass through the byte picture will suffice whereas eight separate passes, one for each peak, would be needed if each peak picture were stored separately.

An *expand* procedure, on the other hand, leaves all image points with value one unaffected. The peak pictures, after having been "shrunk and expanded", will be combined by a "logical OR" to form a binary picture which will be used to create a new direction image, as explained later. Therefore, as soon as the *expand* operation on a pixel representing any one peak results in its value getting changed from zero to one, processing can cease for that pixel.

(7) We repeat the histogramming procedure using only points from the first set of peaks which survived the shrink and expand procedure. A new direction image is created by taking the product of the old direction image with the result of the logical OR after shrinking and expanding. The upper and lower left quadrants of Figure 2e again show the direction picture and histogram before the shrink and expand procedure. The upper and lower right quadrants show them after this process. Note how much sharper the new direction histogram is.

(8) A new set of peaks is now selected from this histogram by the method described previously. The points corresponding to the new strongest four peaks are shown in Figure 2f.

(9) In each peak picture, we now try to locate lines of a specified orientation (according to the direction associated with that peak) plus or minus a small amount (according to the width of the peak). We employ a Hough voting procedure. Thus, we have decomposed the two-dimensional Hough problem into two one-dimensional problems: the first one is to find the approximate orientation of the line, then knowing approximately this direction we use a Hough voting procedure to find the intercept of the line. This last step is done by looking for the largest number of points which fall on the line, then the next largest, and so on until the number of points falls below some percentage of the strongest line. The lines found are shown superposed on the points for each peak in Figure 2f.

(10) Finally, we locate the clusters of points on the lines found, by histogramming the density of points along and near the lines. Clusters are easily detected since the density histogram typically consists of a few peaks with sharp cutoffs between them. Thus the lines are broken into line segments. Figure 2g shows all the line segments that are found superposed on the original picture.

Similar results obtained for other images are shown in Figures 3, 4 and 5, including a "simulator picture" taken from a 96:1 scale model comprised of a road network and a robot arm carrying a camera (cf. [4]).

This bootstrap processing requires about 90 CPU seconds on a VAX 11/785 and is principally spent in the enhancement step and the Hough step; however, the linear enhancement step which requires about a third of this time is often unnecessary. This estimate includes time spent in creating files that will not be needed to actually run the ALV. Besides, during most of the traverse, the ALV will be guided by the *feed-forward* stage, which requires far less processing time.

We expect to reduce the entire process of bootstrapping to several seconds by utilizing pipelined image processing hardware. If this can be achieved, the ALV should be operable at 10 km/hr.

2.2. Segmentation

Methods for region-based segmentation of grey scale and color images were developed. The methods combine edge-preserving smoothing with a connected components (CC) algorithm. Good performance is primarily a result of the efficient smoothing algorithms used, the Symmetric Nearest Neighbor filter (SNN) and its color version (color-SNN). The grey scale filter was introduced by Harwood et al. in [5]. It uses both spatial and nearest-neighbor constraints on image pixels to smooth an image.

For color images, a multiband version was developed. The color-SNN as well as the multiband version of the CC algorithm make use of a new measure of edge information in color images based on histograms of absolute color differences. A difference histogram is generated in one pass through the image by considering absolute differences between a given pixel and its four neighbors. Instead of using the original histogram, a cumulative histogram of differences may be used. A more detailed description of the color image processing methods is given in [6].

As the first step in segmentation, the image is smoothed by the SNN or color-SNN filter. Normally two or three iterations of 3×3 filtering are needed to sharpen edges and smooth homogeneous areas. As an option, the images can be blurred prior to smoothing in order to get a smaller number of

connected regions (i.e., to reduce the effects of background texture and minor variations on the surface of the road). A better way, however, would be to reduce the size of the image if possible. In our experiments, very good segmentations were obtained for road images with 128×128 pixels. To make edges even sharper and to avoid mismerging of regions at some critical points, grey scale images or color bands can be edge-enhanced with a grey-scale filter MINRANGE as described in [6].

After the image is smoothed it is segmented by a two-pass connected components (CC) algorithm, in which two adjacent pixels are said to be connected if the likelihood or frequency of their absolute grey scale or color difference is sufficiently large. The only parameter is a threshold, expressed as a centile of frequencies, which is supplied by the user. First the histogram of absolute grey scale or color differences is computed, and then used to find the desired threshold value.

The two passes of the CC algorithm are the same as those of the standard one. Row by row, pixels are assigned labels by comparing each pixel with the four adjacent pixels above or to the left, which have already been labeled as the image is scanned from top to bottom, left to right. Then, in the second pass, the pixels with component-equivalent labels are relabeled uniquely. For each connected region, different types of information may be collected during the labeling process, such as area, surrounding window and average grey value or color in each band.

Figure 6 shows the results of grey level segmentation for the "INTERSECTION" image as well as superposition of the linear features on the segmented image. Figure 7 shows the red and blue bands of the "STRAIGHT ROAD" image and the results of color segmentation using these bands. Figure 8 shows color segmentation results for the green and blue bands of "INTERSECTION". Figure 9 illustrates grey level segmentation results on a simulator image, "THE HILL", and superposition of linear features on the segmentation.

Our experiments indicate that color segmentation is more reliable and less sensitive to changes in parameter values than the grey scale method, even though the grey scale method performs almost equally well for most of the road images. Good color segmentations were obtained for a set of eight unrelated 128×128 pixel road images using the same set of parameter values for these different images.

Region-based segmentation can be used in different ways in the bootstrap phase. By fitting line segments to the boundaries of those regions which are large enough, and by selecting those line segments which are long enough, a representation compatible with the linear features can be obtained. The selection of the line segments may be constrained in various ways using preknowledge about the scene being analyzed. The sizes, shapes, positions and

colors of segmented regions can also be used to aid the interpretation of the dominant linear features obtained by the method presented in Section 2.1.

A two-band color segmentation of a 128×128 image with three iterations of color-SNN smoothing and one iteration of sharpening requires about 60 CPU seconds on a VAX 11/785, when using non-optimized code. A real-time implementation in hardware should be quite straightforward.

3. FEED-FORWARD IMAGE PROCESSING

Once bootstrapping is complete and the ALV begins to move down the road, one can significantly reduce the image processing load by essentially predicting the approximate location of important features in the scene. This requires that the ALV have a dead-reckoning capability so that it knows how far it has traveled through the world model (created from the last image) by the time the next image is ready to be processed. While the image processing is going on, the ALV is essentially "traveling blind". During this blind travel, the vehicle navigates (conventionally) through a static world model. When it is ready to process the next frame, its perspective of the world has changed though the world itself hasn't. If the ALV knows approximately where it is in relation to the world model, it can predict (i.e., focus visual attention) where in the imagery the important features are, place windows on small portions of the image and locate features with much less effort than was required in the bootstrap mode. Then, imposing road continuity in the world allows road tracking through the image and model updating at much higher rates. This, in turn, allows the ALV to attain higher speeds of travel. If the system finds that it is having trouble in the feed-forward mode of operation, it can slow down (or stop) and resort to bootstrapping once again.

The more accurate the predictions that can be made, the smaller the windows need be in which processing must occur, and thus the faster the vehicle can move. Making accurate predictions requires knowing the location and orientation of the vehicle (and the camera) with respect to its previous position when imagery was taken. This puts constraints on the accuracy of the dead-reckoning system aboard the ALV, as well as the pointing accuracy of the pan and tilt mechanism carrying the camera (no stereo vision is being utilized for this task). Some simple error studies were carried out in order to ascertain the accuracies required and are described in [3,4]. A prediction amounts to placing windows at the bottom of the image (i.e. close to the vehicle) in which the road boundaries are to be found. The complete boundaries are then tracked from the bottom to the top of the image.

3.1. Linear feature extraction

Once bootstrapping is complete, the locations of dominant linear features can be predicted near the bottom of the next image.

The set of image processing steps required to extract the dominant linear features in the *feed-forward* mode is a subset of those required in the *bootstrapping* mode. In particular, no enhancement of linear features is necessary prior to histogramming. As we are looking for a feature that is a line segment of width one pixel, the signal to noise ratio inside the area of the image where we are looking is inversely proportional to the *feature length*.

This implies that for the small windows used in the feed-forward mode, linear feature enhancement is unnecessary. Other differences between the two modes also exist in the detection of the peaks after histogramming and in the Hough transform application.

The various processings steps applied within a window are:

(1) The original picture is artificially blurred to reduce the noise.

(2) The Sobel operator is then applied to obtain the gradient magnitude and direction.

(3) A histogram of the directions is then constructed, weighted by the square of the gradient magnitude. As previously, this histogram is smoothed by a Gaussian filter and the peaks are automatically selected. Then, if the window under consideration is at the bottom of the picture (i.e. placed by the Predictor), the strongest peak is chosen. Otherwise, the single peak chosen is that with direction closest to the line found in the previous window. From this chosen peak, we can then create the corresponding "peak picture".

(4) We now try to locate a line in the peak picture, having a specified orientation plus or minus a small amount. If this is the first window then the lines are found by the same Hough voting procedure that was described for the bootstrapping mode. Processing differs for subsequent windows because of the introduction of an additional constraint; lines found in these windows must be connected to the ones found in the previous windows. This constraint follows naturally from the constraint in the real world of *road continuity*. Finding the line in this case becomes much easier because the intercept of the line is known at one point called the *pivot* (this point is the end of the line found in the previous window). The Hough accumulator contains only the votes for each line direction within the width of the selected peak.

(5) We then retain as road boundary one-half of the line found. This prevents the line from overshooting beyond the actual road boundary.

Figures 10, 11, 12 and 13 illustrate this sequence of operations on four different images. The method is quite able to follow around bending roads, but is currently unable to deal with intersections. Extensions to the method will solve this problem.

Once a window is processed, the next window is chosen in such a way that the length of the line segment found in this extrapolated window is maximized. The window size should also be dynamically adjustable during the feed-forward stage.

The success of feed-forward image processing depends heavily upon the accuracy with which the first window is positioned by the Predictor on the lower part of the image. We can make this initial window placement less

critical by making it larger than the ones that follow it (which track the road out to greater distances). However, larger starting windows imply longer computing times. Processing is automatically stopped when the window is within 20% of the top of the image or when the window leaves the image.

It takes about 9 seconds of CPU time to detect road boundaries in a 255×185 image in the feed-forward mode, which is an improvement by a factor of 10 over the time required for bootstrapping. This algorithm has also been modified to run on a VICOM image processing system, requiring an average of 4 seconds of CPU time per image. Further details on the VICOM implementation are found in [7].

3.2. Segmentation

The segmentation procedure for the feed-forward mode is basically the same as in the bootstrap mode. However, here we can use knowledge about the predicted location of the road to collect statistics of grey scale or color differences on the surface of the road and to use this information to derive a proper threshold for segmentation. Next we describe a simple procedure that was developed for non-patchy roads, i.e., for roads that are segmented into one major region (with many small embedded patches possible).

(1) The original image is smoothed as in the bootstrap mode.

(2) A window is positioned at the lower middle part of the road.

(3) A histogram of grey scale or color differences in the window is computed. A threshold is determined that would connect a given percentage of all 8-connected pixel pairs in the window.

(4) The entire image is segmented using that threshold value.

(5) The major road segment is found by counting the number of different region labels in the same window that was used for threshold selection.

(6) Line segments are fit to the border of the chosen segment.

If the surface of the road is patchy due to significant color differences on the surface of the road or due to shadows, the road can be segmented into several regions. In this case a more general approach should be used, e.g., in which line segments are fit to the borders of the major regions and only those line segments are chosen that satisfy given length and direction criteria. Figures 14 and 15 show some results of grey segmentation in the feed-forward mode.

4. CONCLUDING REMARKS

We have presented several image processing algorithms that were developed for use by an autonomous land vehicle in order to navigate roadways. Dominant linear features or regions can be extracted in two different modes, a bootstrap mode where no prior visual information is available, and a feed-forward mode used when results of a prior road image combined with dead-reckoning allow prediction of the approximate location of important road features in subsequent images. These algorithms have been implemented and tested on a variety of road images including "simulator images" obtained from a scale model road network. Our current efforts consist of speeding up these algorithms, and making them more flexible. In particular, it would be useful to have a greater degree of "top-down" control over the Image Processing Module by the Reasoning Module.

The next "generation" of the system should be able to combine these different kinds of representations, linear features and grey-level or color regions, to give a more complete interpretation of the scene (cf. [3,4,8]). In addition, the system should be able to switch between the global processing of the bootstrap mode, and the local processing of the feed-forward mode, in order to overcome problems that may be encountered in such cases as intersections, sharp bends and shadows cast across a road.

We thank Mark Westling and David Harwood for making available to us their novel segmentation algorithm.

References

[1] "Strategic Computing", Defense Advanced Research Projects Agency report, October 28, 1983; prepared by BDM Corp. (McLean, VA).

[2] L.S. Davis, A. Rosenfeld, and A.M. Waxman, "Visual Ground Vehicle Navigation", Final Report, Workshop on Autonomous Ground Vehicles, Leesburg, VA, October 24 - 26, 1983.

[3] A.M. Waxman, J. Le Moigne, and B. Srinivasan, "Visual Navigation of Roadways", 1985 IEEE International Conference on Robotics and Automation, pp. 862-867, St. Louis, MO, March 1985.

[4] A.M. Waxman, J. Le Moigne, L.S. Davis, E. Liang, and T.K. Siddalingaiah, "A Visual Navigation System for Autonomous Land Vehicles", University of Maryland, Center For Automation Research Technical Report 139, July 1985.

[5] D. Harwood, M. Subbarao, H. Hakalahti, and L.S. Davis, "A New Class of Edge-Preserving Smoothing Filters", University of Maryland, Center for Automation Research Technical Report 59, May 1984.

[6] M. Pietikainen and D. Harwood, "Edge Information in Color Images Based on Histograms of Differences", University of Maryland, Center for Automation Research Technical Report 112, March 1985.

[7] L.S. Davis and T. Kushner, "Road Boundary Detection for Autonomous Vehicle Navigation" , University of Maryland, Center For Automation Research Technical Report 140, July 1985.

[8] J. Le Moigne, "Domain-dependent Reasoning for Visual Navigation of Roadways", University of Maryland, Center For Automation Research Technical Report, in preparation.

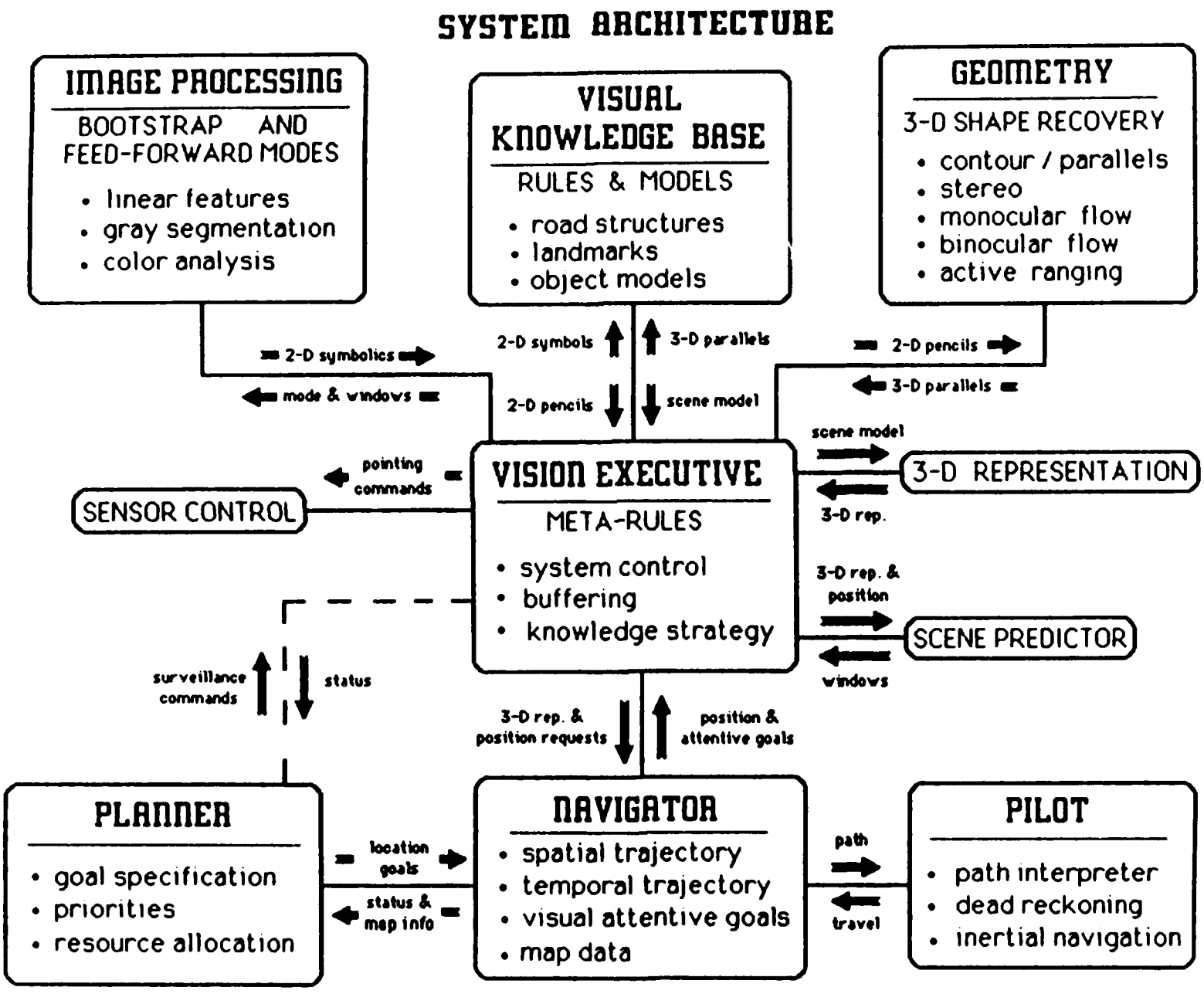


FIGURE 1. SYSTEM ARCHITECTURE AND ROAD FOLLOWING PROCESS CONTROL

FIGURE 2a.

FIGURE 2b.

FIGURE 2c.

FIGURE 2.
Bootstrap Image Processing steps
for linear feature extraction

Figure 2a. Original image
2b. Upper left : Sobel gradient magnitude
Upper right: Sobel gradient direction
Lower left : results of linear enhancement
Lower right: weighted histogram of the direction image
2c. The points in the original image corresponding
to the strongest four peaks

FIGURE 2d.

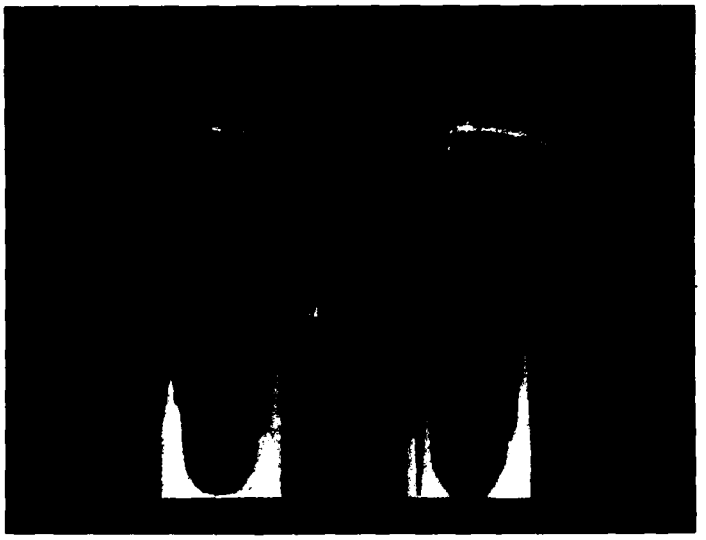


FIGURE 2e.

FIGURE 2f.

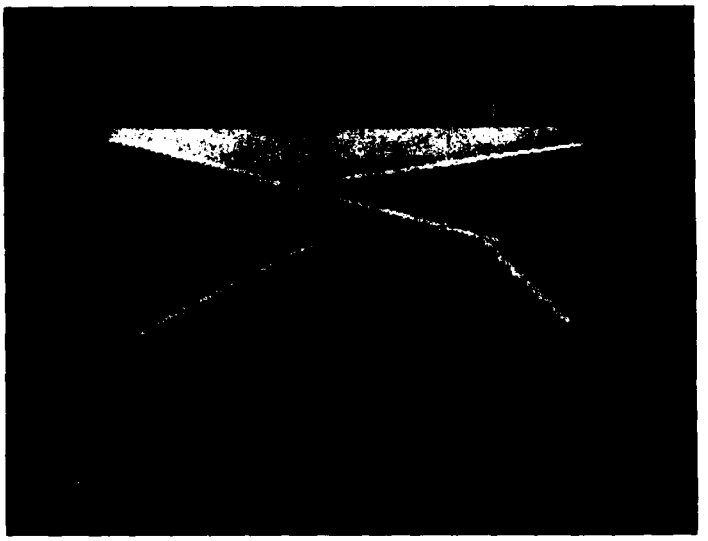


FIGURE 2g.

FIGURE 2. (CONTINUED)

Figure 2d. The points which survived the shrinking and expanding
2e. Upper left : Direction image before shrinking and expanding
Lower left : Corresponding weighted histogram
Upper right: Direction image after shrinking and expanding
Lower right: Corresponding weighted histogram
2f. Lines superposed on peaks obtained from the new histogram
2g. Lines superposed on the original picture

FIGURE 3.

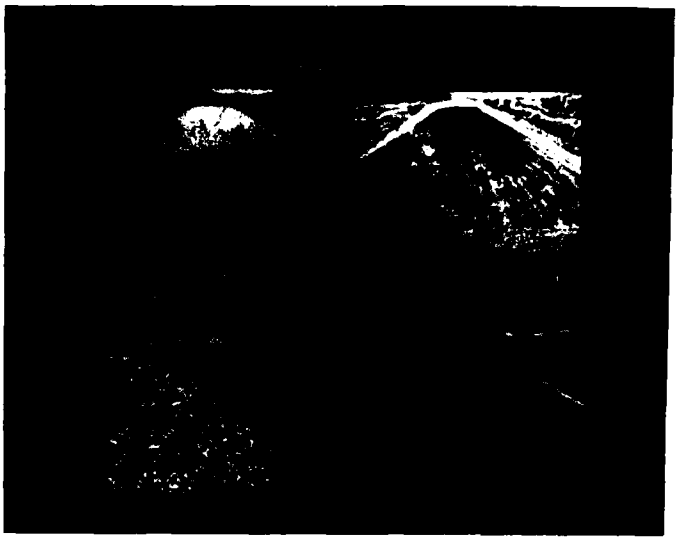


FIGURE 4.

FIGURE 5.

Other examples of bootstrap processing for linear feature extraction

Figure 3. Intersection
4. Straight road
5. Simulator image: the Hill

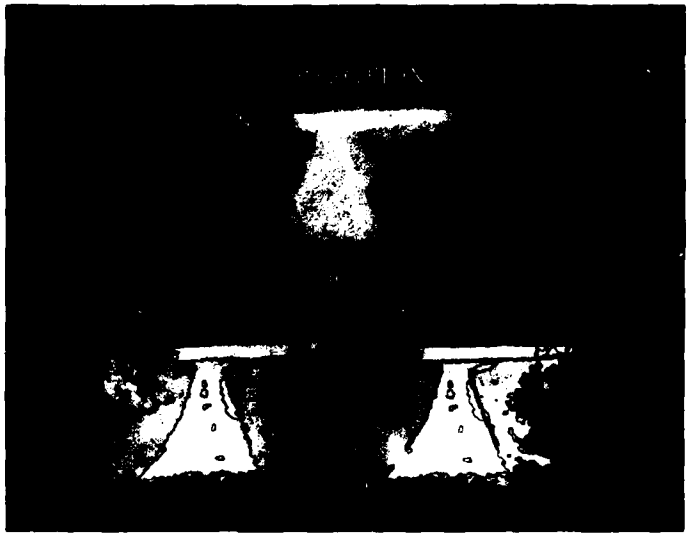


FIGURE 6.

FIGURE 7.

FIGURE 8.

FIGURE 9.

Results of Bootstrap Segmentation

Figure 6. Grey level segmentation of "Intersection"
Upper image : original
Lower left : grey segmentation
Lower right: superposition of linear features on grey segmentation
7. Color segmentation of "Straight road"
Upper left and right : red and blue band images
Lower image : color segmentation
8. Color segmentation of "Intersection"
Upper left and right : green and blue band images
Lower image : color segmentation
9. Grey level segmentation of a simulator image: "the Hill"
Upper image : original
Lower left : grey segmentation
Lower right: superposition of linear features on grey segmentation

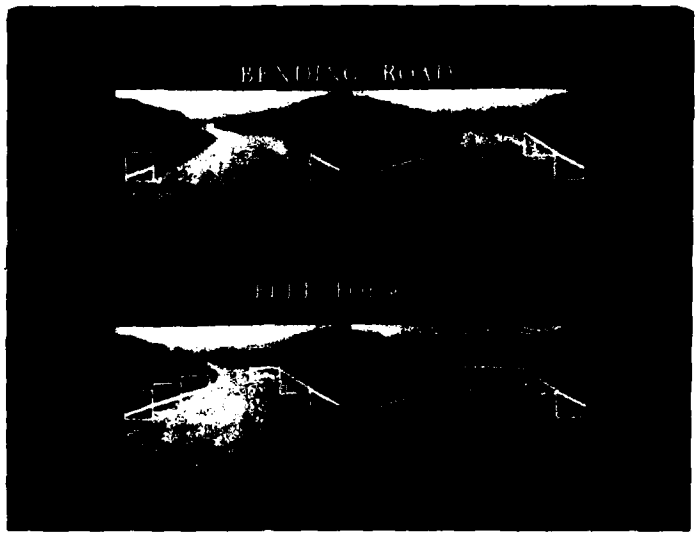


FIGURE 10.

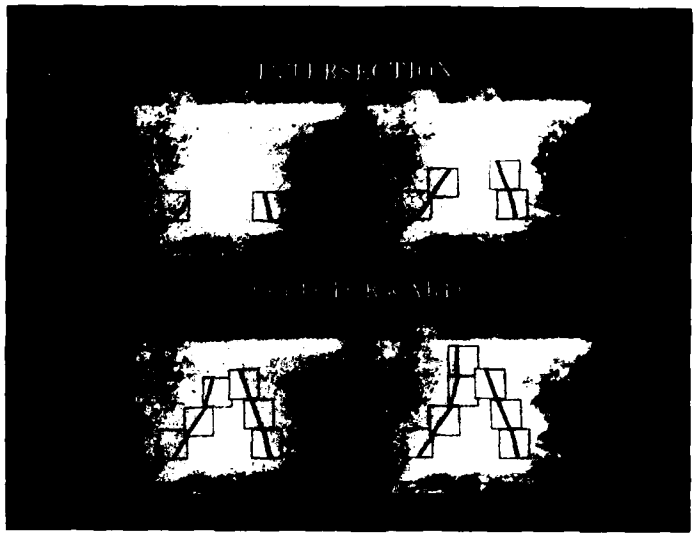


FIGURE 11.

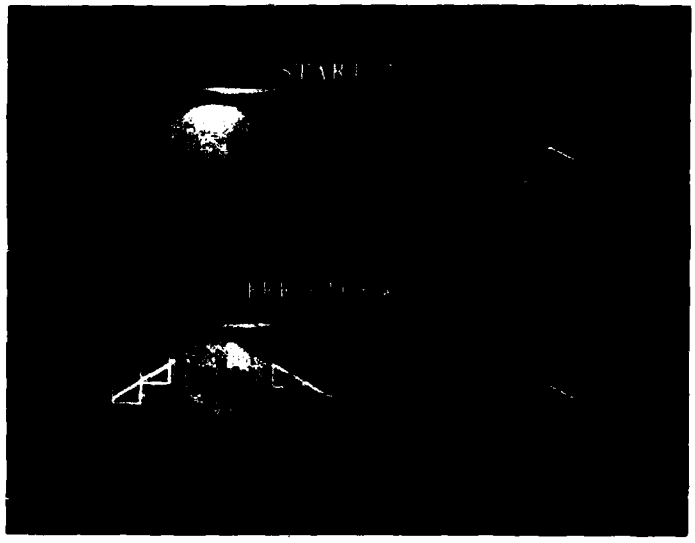


FIGURE 12.

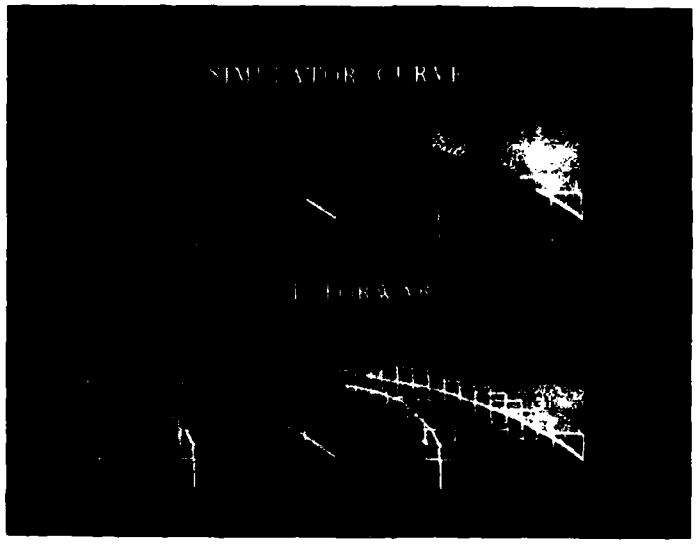


FIGURE 13.

FEED-FORWARD IMAGE PROCESSING.
Successive steps of linear feature extraction in the feed-forward mode on four different examples

Figure 10. Bending road
11. Intersection
12. Straight road
13. Simulator image: the Curve

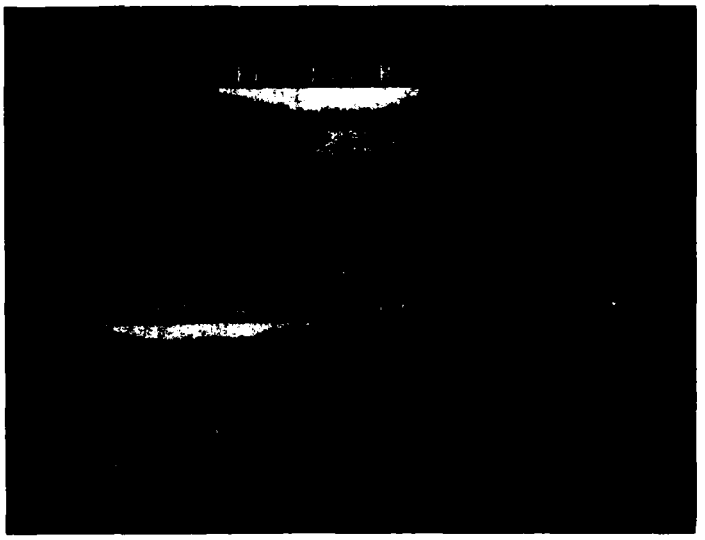


FIGURE 14

FIGURE 15

FEED-FORWARD IMAGE PROCESSING (CONTINUED)

Successive steps of grey level segmentation in the feed-forward mode on two different examples

Figure 14. Bending road
15. Simulator image: the Curve

END

FILMED

12-85

DTIC